Optimal scheduling of parallel processing systems with real-time constraints

François Baccelli, Zhen Liu, Don Towsley

INRIA

UNITE DE RECHERCHE
INRIA-SOPHIA ANTIPOLIS

Institut National
de Recherche
en Informatique
et en Automatique

Domaine de Voluceau
Rocquencourt
BP 105
78153 Le Chesnay Cedex
France
Tél. (1) 39 63 55 11

Rapports de Recherche

N° 1113

Programme 3
Réseaux et Systèmes Répartis

OPTIMAL SCHEDULING OF PARALLEL PROCESSING SYSTEMS WITH REAL-TIME CONSTRAINTS

François BACCELLI
Zhen LIU
Don TOWSLEY

Novembre 1989

RR.1113

Optimal Scheduling of Parallel Processing Systems with Real-Time Constraints
Ordonnancement Optimal dans les Systèmes Parallèles avec Contraintes Temps Réel

François BACCELLI, Zhen LIU

INRIA
Centre Sophia Antipolis
06565 Valbonne Cedex
France

and

Don TOWSLEY*

Dept. of Computer & Information Science
University of Massachusetts
Amherst, MA 01003
U.S.A.

June 1989

*The work of this author was supported by NSF under contract ASC 88-8802764 and ONR under contract N00014-87-K-0796.

Abstract

We consider parallel execution of structured jobs with real time constraints in (possibly heterogeneous) multiprocessor systems. A job is composed of a set of tasks and a partial order specifying the precedence constraints between the tasks. The task processing times are random variables with known probability distribution functions. The interarrival times of the jobs are also random variables with arbitrary distributions. The real time constraints are specified by reference times, also called soft real-time deadlines. In the discussion we assume first that all the jobs have the same task graph, i.e., the same task set and the same partial order. We assume that there is a predefined mapping from the set of tasks onto the set of machines, identical for all jobs, that allocates tasks to machines. We focus on dynamic scheduling policies which do not use information on the processing times of the tasks to be scheduled. The policies can be non-preemptive or preemptive-resume. For non-preemptive policies, we assume that task processing times are independently and identically distributed, whereas for the preemptive ones, we assume that they are independently and identically distributed and come from some specific distributions. Some of the results that are obtained generalize to the case where jobs have a random structure. This more difficult case is treated at the end of the paper.

We are interested in such performance criteria as the number of jobs in the system, the throughput in term of jobs and the job lag times, which correspond to the differences between the reference times and the completion times of tasks or jobs. We show that FCFS policies, applied at task level (which imply FIFO for jobs), stochastically minimize the number of jobs and maximize the system throughput. We establish that FCFS policies minimize the vector of the transient response times in the sense of Schur convex ordering, and minimize the stationary response times in the convex ordering sense. As to real time applications, we prove that within the class of local order preserving policies that is defined in the paper, the SRF (shortest reference first) and LRF (longest reference first) policies bound respectively from below and from above the lag times vector of the n first jobs, in the Schur convex sense. This in turn yields an analogous convex ordering among the stationary job lag times.

Keywords: Parallel Processing, Precedence Constraints, Real-Time Constraints, Throughput, Lag Time, Response Time, Optimal Scheduling, Extremal Policy, FCFS, LCFS, FIFO, LIFO, Local Order Preserving, Shortest Reference First, Longest Reference First, Stochastic Ordering, Convex Ordering, Schur Convex Ordering, Convex Symmetric Ordering.

Résumé

On considère l'exécution parallèle de programmes structurés en graphes de tâches sur des systèmes multiprocesseurs soumis à des contraintes temps réel. Un programme est composé d'un ensemble de tâches et d'un ordre partiel spécifiant les contraintes de précédence entre les tâches. Les longueurs des tâches sont des variables aléatoires, de fonctions de répartition connues. Les dates d'arrivée des programmes forment un processus ponctuel aléatoire de loi arbitraire. Les contraintes temps réel sont spécifiées par les dates de référence, également appelées *échéances temps réel.* On suppose dans un premier temps que tous les programmes ont le même graphe de tâches (c'est-à-dire, le même ensemble de tâches et le même ordre partiel) et qu'il y a une affectation prédéfinie de l'ensemble des tâches sur l'ensemble des machines, identique pour tous les programmes. On se concentre sur les politiques d'ordonnancement dynamiques qui n'utilisent pas d'informations sur les temps de calcul des tâches à ordonner. Les politiques peuvent être non-préemptives ou préemptives avec reprise. Pour les politiques non-préemptives, on suppose que les longueurs des tâches sont indépendamment et identiquement distribuées. Dans le cas des politiques préemptives, on suppose de plus que leurs fonctions de répartition sont exponentielles. Une partie des résultats obtenus se généralise au cas où les programmes ont une structure aléatoire, traité à la fin de l'article.

On cherche à optimiser les critères de performance suivants: le nombre de programmes dans le système, le débit des programmes, et le *retard* des programmes, où ce retard est défini comme la différence entre l'échéance et la date de fin d'exécution du programme. On démontre que la politique PAPS (Premier Arrivé Premier Servi), appliquée au niveau de tâches (qui implique PEPS (Premier Entré Premier Sorti) pour les programmes), minimise le nombre de programmes en attente dans le système et maximise le débit pour l'ordre stochastique fort. On établit aussi que la politique PAPS minimise le vecteur transitoire des temps de réponse pour l'ordre Schur convexe, et minimise le temps de réponse stationnaire pour l'ordre convexe. Pour les applications avec des contraintes temps réel, on prouve que dans la classe des politiques *préservant localement l'ordre d'exécution* (définie dans l'article), les politiques PEA (Première Echéance d'Abord) et DEA (Dernière Echéance d'Abord) encadrent le vecteur transitoire des retards pour l'ordre Schur convexe, ce qui entraîne un encadrement analogue du retard stationnaire pour l'ordre convexe.

Mots-Clés: Informatique parallèle, contraintes de précédence, contraintes temps réel, files d'attente, débit, temps de réponse, retard, ordonnancement optimal, politique extrémale, ordonnancement stochastique, ordre convexe, ordre Schur convexe.

1 Introduction

We are concerned with scheduling policies in a parallel processing system with real time constraints. In the system, there are $K \geq 1$ (possibly heterogeneous) machines. A job of the system is composed of a set of sequential tasks and a partial order, described by a task graph, specifying the precedence constraints between the executions of the tasks. The task processing times are random variables with known probability distribution functions. The jobs arrive at random times, with arbitrarily distributed of interarrival times. The real time constraints are specified by reference times, also called soft real-time deadlines. The jobs may all have the same task graph, i.e., the same task set and the same partial order, or some random task graph to be specified.

We assume that there is a predefined mapping from the set of tasks onto the set of machines, identical for all jobs, that allocates tasks to machines. In the system, each machine has a queue of tasks. When a job arrives to the system, it is immediately split into tasks which are queued up in accordance with the mapping. When a machine is free, it chooses one of the enabled tasks from its queue for processing where a task is said to be enabled if all of its predecessors belonging to the same job are completed. Within this context, a scheduling policy is non-idling if a machine is never free unless there are no enabled tasks in its queue. In this paper, we consider idling policies as well as non-idling policies. It will be assumed that the mapping from the task set onto the machine set does not depend on the processing times of the tasks.

We focus on dynamic scheduling policies which do not use information on the processing times of the tasks to be scheduled. The policies can be non-preemptive or preemptive-resume. For the non-preemptive policies, we assume that task processing times are independent and identically distributed (i.i.d.) random variables, whereas for preemptive policies, we assume that they have certain specific distribution functions.

The performance criteria of interest are the number of jobs in the system, the throughput in term of jobs, the job lag times (defined to be the difference between the job departure time and its reference time) and the response times of tasks and jobs. In real time systems, a job turns *bad* if its completion time is larger than its reference time, or equivalently if the lag time is positive. The probability that a job turns bad is an important performance criterion in such systems.

Some earlier studies on related matters are available in the context of classical queueing

systems. The convex ordering properties of the FIFO and LIFO (Last In First Out) service policies have been provided, by Doshi and Lipper [5], Shanthikumar and Sumita [8], and Vasicek [10], for the stationary sojourn time in $M/G/1$ queues, the stationary sojourn time in $G/G/1$ queues, and the stationary waiting time in $G/G/c$ queues, respectively. As to real time constraints, the optimality of the smallest reference time (SRF) policy was first established in [9] for queues in tandem.

Our results generalize most of these earlier results to the general model of parallel processing systems that was initially introduced and analyzed in Baccelli and Liu [1]. In addition, the Schur convex ordering property that is established for the transient response times or lag times vectors appears to be new, even for classical queues.

The paper is organized as follows. We assume first that all the jobs have a same logical structure. In Section 2 we define the system model and the notation used in the paper, and the scheduling policies as well as the performance criteria under consideration. In Section 3, we provide the requisite notions of stochastic ordering. In Section 4, we consider the case where only such performance criteria as the number of jobs, the throughput, and the response times of tasks or of jobs. We establish first a set of results on the optimality of the First Come First Serve policy. In Section 5, we focus on the job lag times. We show some extremal properties of the Shortest Reference First and of the Longest Reference First policies over the class of Local Order Preserving policies, to be defined in due time. In Section 6, we provide further results on these extremal policies in the stationary regime. In Section 7, we show that all these results hold for preemptive policies, provided the task processing times have some special distributions. In Section 8, we discuss the case where the jobs have a random structure. It is first established that all the results concerning the class of Local Order Preserving policies still hold. The optimality properties of First Come First Serve policies still hold for some appropriately defined FCFS policies. In Section 9, we describe some applications of our results to specific queueing networks. Finally Section 10 summarizes the results obtained in this paper.

2 Notation and Definitions

2.1 The Parallel Processing System Model

The parallel processing system consists of $K \geq 1$ (possibly heterogeneous) machines. These machines can communicate with each other either via some communication link or via access to some common memory. The jobs are parallel programs with a similar logical,

possibly random structure. We focus first on the former case. The latter will be discussed in Section 8.

The jobs arrive to the system at the times $a_1 = 0 < a_2 < \cdots < a_n < \cdots$, and are numbered $1, 2, \cdots$. Every job brings a set of (sequential) tasks which can be described by the same task graph $G = (V, E)$, where $V = \{1, 2, \cdots, |V|\}$ is the set of vertices corresponding to the tasks $\{T_1, T_2, \cdots, T_{|V|}\}$ of the job and E the set of directed edges indicating precedence relations between tasks. For all $i, j \in V$, relation $(i, j) \in E$ implies that task T_j has to wait for task T_i to be completed before being processed.

We assume that G is acyclic, and, without loss of generality, we assume that the vertices in the task graph are ordered in such a way that $(i, j) \in E$ implies $i < j$ (such an ordering is always possible in an acyclic graph).

Define the set of immediate predecessors of vertex i, $p(i)$, to be the set of vertices that have a direct link to vertex i

$$p(i) = \{j \in V | (j, i) \in E\}$$

and the set of immediate successors of queue i, $s(i)$, to be the set of vertices to which i has a direct link

$$s(i) = \{j \in V | (i, j) \in E\}.$$

We define the set of immediate predecessors for a set of vertices, $S \subseteq V$ to be the union of the immediate predecessor sets of the vertices in S,

$$p(S) = \bigcup_{i \in S} p(i).$$

Similarly, $s(S)$ is the set of immediate successors for the set of vertices S,

$$s(S) = \bigcup_{i \in S} s(i).$$

It is assumed that G is connected. The task processing times are random variables denoted by $\{s_{i,n}\}_{n=1}^{\infty}$, $1 \leq i \leq |V|$, where the first index (i) indicates the task and the second one (n), the job; i.e., task T_i of job n (which will also be denoted by $T_{i,n}$) requires $s_{i,n}$ units of processing time. The interarrival of the jobs are also random variables and denoted by the sequence $\{u_n = a_{n+1} - a_n\}_{n=1}^{\infty}$.

Associated with the n-th job, there is a *reference time* (which is also referred to as a *soft real-time deadline* in the literature on real time computing systems) denoted by

d_n, $n = 1, 2, \cdots$. This reference time is used to evaluate the system performance. For instance, the fraction of jobs that complete before their reference times is an important quality criterion for this type of systems. The *relative reference time*, denoted by r_n, is the difference between the reference time and the arrival time: $r_n = d_n - a_n$, $n = 1, 2, \cdots$.

All the results of the paper are based on the following set of assumptions.

Assumption *A:* $\{s_{i,n}\}_{n=1}^{\infty}$, $1 \leq i \leq |V|$, *are mutually independent i.i.d. sequences of RV's, and independent of the sequences* $\{u_n\}_{n=1}^{\infty}$ *and* $\{r_n\}_{n=1}^{\infty}$.

2.2 Scheduling Policies

There is a static mapping $\mathcal{M}$ from the set of tasks onto the set of machines, that assigns tasks to machines. We assume that $\mathcal{M}$ is the same for all jobs. For all $i \in V$, $\mathcal{M}(i)$, denotes the index of the machine on which task T_i is allocated.

Each machine has a queue of tasks. When a job arrives to the system, it is immediately split into tasks which are allocated to the K machines in accordance with the mapping $\mathcal{M}$. When a machine is free, it chooses one of the enabled tasks from its queue for processing. A task is said to be enabled if all of its predecessors belonging to the same job are completed. Throughout this paper, we limit ourselves to the class of dynamic scheduling policies which do not use information on the processing times of the tasks to be scheduled.

Within this context, a scheduling policy is non-idling if a machine is never free unless there are no enabled tasks in its queue. In this paper, we consider idling policies as well as non-idling policies. A scheduling policy is non-preemptive if the execution of a task cannot be interrupted once it has begun execution. A scheduling policy is preemptive-resume if the execution of a task can be interrupted by another task, and if this occurs, the first task is then resumed from the preemption point. In this paper, we assume that task $T_{i,n}$ ($1 \leq i \leq N$, $n \geq 0$) can only be preempted by the tasks of the type $T_{i,m}$, $m \neq n$. Denote by Π (resp. Π') the class of non-preemptive (resp. preemptive) scheduling policies. Clearly $\Pi \subset \Pi'$.

Let $b_{i,n}(\pi)$, $c_{i,n}(\pi)$, where $1 \leq i \leq |V|$, $n \geq 1$, respectively denote the times when task $T_{i,n}$ is attended and completed, according to policy π. In case the scheduling is allowed to be preemptive, $b_{i,n}(\pi)$ refers to the time when the task $T_{i,n}$ is attended for the first time, and $c_{i,n}(\pi)$ to the time when $T_{i,n}$ is completed and definitively leaves the system. Denote

by $c_n(\pi)$, $n \geq 1$, the departure time of the n-th job. By definition,

$$c_n(\pi) = \max_{1 \leq i \leq |V|} c_{i,n}(\pi), \qquad n \geq 1.$$

A non preemptive policy π, is said to be feasible if it satisfies the following three conditions:

1. π starts attending a task only after it has arrived: for all i, n, where $1 \leq i \leq |V|$, $1 \leq n \leq m$,

$$b_{i,n}(\pi) \geq a_n \tag{2.1}$$

2. A machine processes at most one task at a time: for all i, n, there exist no j, l, $1 \leq i, j \leq |V|$, $1 \leq n, l \leq m$, such that

$$\mathcal{M}(i) = \mathcal{M}(j), \quad \text{and} \quad b_{i,n}(\pi) < b_{j,l}(\pi) < c_{i,n}(\pi) \tag{2.2}$$

3. The precedence relations are satisfied: for all i, n, where $1 \leq i \leq |V|$, $1 \leq n \leq m$,

$$b_{i,n}(\pi) \geq \max_{j \in p(i)} c_{j,n}(\pi) \tag{2.3}$$

One of the most commonly used classes of scheduling policies is First Come First Serve (FCFS) which requires that for all i and n, $1 \leq i \leq |V|$, $n \geq 1$, task $T_{i,n+1}$ be processed after $T_{i,n}$ has been completed. Note that such policies imply FIFO for jobs. Nevertheless, it may happen that under a FCFS policy, a machine processes $T_{j,n+1}$ before $T_{i,n}$ for some $j \neq i$. We use Ξ (resp. Ξ') to denote the class of non-preemptive (resp. preemptive) FCFS policies.

2.3 Performance Criteria

The performance criteria under consideration are the following.

1. $N_t(\pi)$, number of jobs in the system at time t under policy π:

$$N_t(\pi) = \sum_{n=1}^{\infty} \mathbf{1}_{\{a_n \leq t \bigwedge c_n > t\}}.$$

2. $\mathcal{T}_t(\pi)$, throughput of the system by time t, i.e., number of jobs having left the system by time t under scheduling π divided by t:

$$\mathcal{T}_t(\pi) = \left(\sum_{n=1}^{\infty} \mathbf{1}_{\{c_n \leq t\}}\right)/t.$$

3. $L_n(\pi)$, lag time of the n-th job under π:

$$L_n(\pi) = c_n(\pi) - d_n.$$

4. $R_n^i(\pi)$, response time of task $T_{i,n}$ under π

$$R_n^i(\pi) = c_{i,n}(\pi) - a_n.$$

5. $R_n(\pi)$, response time of the n-th job under π

$$R_n(\pi) = c_n(\pi) - a_n.$$

When these random variables converge weakly when $t \to \infty$ or $n \to \infty$, we denote by $N(\pi)$, $\mathcal{T}(\pi)$, $L(\pi)$, $R^i(\pi)$, $R(\pi)$ random variables distributed according to the limiting distributions of $N_t(\pi)$, $\mathcal{T}_t(\pi)$, $L_n(\pi)$, $R_n^i(\pi)$, $R_n(\pi)$, respectively.

Observe that the job response time is a particular case of job lag time where reference times are set to arrival times.

3 Preliminaries on Stochastic Ordering

In this section, we introduce some partial orderings on $I\!R^n$ that are used for deriving our main results.

For x and y in $I\!R^n$, $\leq$ will denote the *coordinatewise* partial ordering defined by

$$x \leq y \quad \text{iff} \quad x_1 \leq y_1, \cdots, x_n \leq y_n.$$

For x and y as above with $\sum x_i = \sum y_i$ and let i (resp. j) be a permutation of the indices $1, \cdots, n$ such that $x_{i(1)} \leq x_{i(2)} \leq \cdots \leq x_{i(n)}$ (resp. $y_{j(1)} \leq y_{j(2)} \leq \cdots \leq y_{j(n)}$). The *majorization* partial semiordering $\prec$ is defined by $x \prec y$ iff

1. $\sum_{k=l}^{n} x_{i(k)} \leq \sum_{k=l}^{n} y_{j(k)}, \quad l = 2, \cdots, n,$

2. $\sum_{k=1}^{n} x_{i(k)} = \sum_{k=1}^{n} y_{j(k)}$.

A function $f : I\!R^n \to I\!R^m$ is said to be $\leq$ (resp. $\prec$) -monotone non-decreasing if $x \leq y$ coordinatewise (resp $x \prec y$) in $I\!R^n$ implies $f(x) \leq f(y)$ coordinatewise (resp. $f(x) \prec f(y)$) in $I\!R^m$. Similarly, one can define $\leq$ (resp. $\prec$) -monotone non-increasing functions to be $f : I\!R^n \to I\!R^m$ such that $x \leq y$ coordinatewise (resp $x \prec y$) in $I\!R^n$ implies $f(x) \geq f(y)$ coordinatewise (resp. $f(x) \succ f(y)$) in $I\!R^m$.

A $\prec$-monotone non-decreasing (resp. non-increasing) function is also called *Schur convex* (resp. *Schur concave*). For instance, $f(x) = \sum x^i$ or $f(x) = \max x^i$ are Schur convex functions.

Let $\mathcal{D}(I\!R^n)$ denote the space of distribution functions on $I\!R^n$ and $\mathcal{F}$ be a set of borel mappings from $I\!R^n$ onto $I\!R$. Consider the binary relation $\leq_{\mathcal{F}}$ on $\mathcal{D}(I\!R^n)$ defined by

$$F \leq_{\mathcal{F}} G \quad \text{iff} \quad \int_{I\!R^n} f(x)F(dx) \leq \int_{I\!R^n} f(x)G(dx) \quad \forall f \in \mathcal{F}$$

such that the integrals are well defined. Two random vectors $X, Y \in I\!R^n$, with F, G as their distributions, are ordered by $\leq_{\mathcal{F}}$,

$$X \leq_{\mathcal{F}} Y \qquad \text{iff} \qquad F \leq_{\mathcal{F}} G.$$

The following instances of sets $\mathcal{F}$ of functions will be considered in this paper

$$\{st\} = \{f : I\!R^n \to I\!R \mid f \text{ monotone non} - \text{decreasing } \}. \tag{3.1}$$

The set $\{st\}$ generates the so called *strong* partial ordering, denoted by $\leq_{st}$, on $\mathcal{D}(I\!R^n)$. The set

$$\{cx\} = \{f : I\!R^n \to I\!R \mid f \text{ convex } \}. \tag{3.2}$$

where the convexity of the function $f : I\!R^n \to I\!R$ is understood in the usual sense, namely the inequality $f(\lambda x + (1-\lambda)y) \leq \lambda f(x) + (1-\lambda)f(y)$ should hold for all λ in the interval $[0,1]$ and every pair of vectors x, y in $I\!R^n$. The set $\{cx\}$ generates the *convex* partial ordering, denoted by $\leq_{cx}$, on $\mathcal{D}(I\!R^n)$.

The set

$$\{scx\} = \{f : I\!R^n \to I\!R \mid f \text{ Schur convex } \} \tag{3.3}$$

defines the *Schur convex* partial ordering, denoted by $\leq_{scx}$, on on $\mathcal{D}(I\!R^n)$.

The set

$$\{cxs\} = \{f : I\!R^n \to I\!R \mid f \text{ convex and symmetric } \} \tag{3.4}$$

defines the *convex symmetric* partial ordering, denoted by $\leq_{cxs}$, on $\mathcal{D}(I\!R^n)$.

If one replaces the convex functions by concave functions in the above definitions, one gets the *concave*, *Schur concave* and *concave symmetric* partial orderings, which are respectively denoted by $\leq_{cv}$, $\leq_{scv}$, $\leq_{cvs}$, $\leq_{icv}$.

One notes immediately that

$$X \leq_{cx} Y \quad \Leftrightarrow \quad Y \leq_{cv} X. \tag{3.5}$$

$$X \leq_{scx} Y \quad \Leftrightarrow \quad Y \leq_{scv} X. \tag{3.6}$$

$$X \leq_{cxs} Y \quad \Leftrightarrow \quad Y \leq_{cvs} X. \tag{3.7}$$

The remainder of this section describes properties concerning the majorization $\prec$:

Theorem 3.1 *The following properties are equivalent*

i $x \prec y$;

ii x *lies in the convex hull of the points* $\{y_\gamma,\ \gamma \in \Gamma\}$, *where* $y_\gamma = (y_{\gamma(1)}, \cdots, y_{\gamma(n)})$ *and* Γ *is the symmetric group of order* n;

iii $\phi(x) \leq \phi(y)$ *for all convex symmetric function* $\phi : I\!R^n \to I\!R$, *where* ϕ *symmetric means* $\phi(x) = \phi(x_\gamma)$ *for all* $\gamma \in \Gamma$.

Proof.

i $\Rightarrow$ *ii*. See [6] and [7].

ii $\Rightarrow$ *iii*. If x lies in the convex hull of $\{y_\gamma,\ \gamma \in \Gamma\}$, then

$$x = \sum_{\gamma \in \Gamma} p_\gamma y_\gamma$$

where the real numbers p_γ are non-negative and sum up to one. If ϕ is symmetric and convex, we have then

$$\phi(y) = \sum_{\gamma \in \Gamma} p_\gamma \phi(y_\gamma) \geq \phi(\sum_{\gamma \in \Gamma} p_\gamma y_\gamma) = \phi(x).$$

iii $\Rightarrow$ *i*. The functions

$$\max_{1 \leq i \leq n} x_i, \quad \max_{1 \leq i,j \leq n} x_i + x_j, \quad \cdots, \quad \max_{1 \leq i_1, \cdots, i_{n-1} \leq n} \sum_{k=1}^{n-1} x_{i_k}, \quad \sum_{i=1,\cdots,n} x_i$$

are all symmetric and convex. ∎

Corollary 3.2

$$X \leq_{scx} Y \quad \Rightarrow \quad X \leq_{cxs} Y. \tag{3.8}$$

Let y be a vector containing n real numbers. We introduce the notation $y_\gamma = (y_{\gamma(1)}, \cdots, y_{\gamma(n)})$ where γ is a permutation on $(1, 2, \cdots, n)$. Let x be another vector containing n real numbers, then $y - x = (y_1 - x_1, \cdots, y_n - x_n)$.

Corollary 3.3 *Let $x_1 \leq x_2 \leq \cdots \leq x_n$ and $y_1 \leq y_2 \leq \cdots \leq y_n$ be real numbers. For any permutation γ on $(1, 2, \cdots, n)$, if there are some i, j, $1 \leq i < j \leq n$, such that*

$$\gamma(i) = k_1, \qquad \gamma(j) = k_2, \qquad \text{and} \quad k_1 > k_2,$$

and if γ' is another permutation of $(1, 2, \cdots, n)$ which is obtained from γ by interchanging the values of γ on i and j:

$$\gamma'(i) = \gamma(j), \quad \gamma'(j) = \gamma(i), \quad \text{and} \quad \gamma'(k) = \gamma(k), \quad k \neq i, \quad k \neq j$$

then

$$(y_{\gamma'} - x) \prec (y_\gamma - x).$$

Proof. By the definition of γ, $i < j$ and $k_1 > k_2$, which imply that $y_{\gamma(i)} - y_{\gamma(j)} \geq 0$ and $x_i - x_j \leq 0$. Thus there exists a real number $0 \leq \epsilon \leq 1$ such that

$$\epsilon(y_{\gamma(i)} - y_{\gamma(j)}) + (1 - \epsilon)(x_i - x_j) = 0.$$

This relation can be rewritten in the two equivalent forms

$$y_{\gamma'(i)} - x_i = y_{\gamma(j)} - x_i = (1 - \epsilon)(y_{\gamma(j)} - x_j) + \epsilon(y_{\gamma(i)} - x_i)$$

and

$$y_{\gamma'(j)} - x_j = y_{\gamma(i)} - x_j = \epsilon(y_{\gamma(j)} - x_j) + (1 - \epsilon)(y_{\gamma(i)} - x_i)$$

which imply that $(y_{\gamma'} - x)$ lies in the convex hull of the points $\{(y_\gamma - x), \gamma \in \Gamma\}$. This shows that $(y_{\gamma'} - x) \prec (y_\gamma - x)$ in view of Theorem 3.1. ∎

As an immediate application of Corollary 3.3, we get

Corollary 3.4 *Let $x_1 \leq x_2 \leq \cdots \leq x_n$ and $y_1 \leq y_2 \leq \cdots \leq y_n$ be real numbers. Then, for all $\gamma \in \Gamma$*

$$(y - x) \prec (y_\gamma - x) \prec (y_\alpha - x) \tag{3.9}$$

where α is defined by $\alpha(i) = n - i + 1,\ i = 1,n$.

We introduce now a partial ordering on Γ which will be of use in the next section. Let $\{x_i\}_{i=1}^n$ be a sequence of mutually different real numbers. Define the binary relation $\prec_{\{x_i\}}$ on the symmetric group Γ as follows:

1. $\gamma' \prec_{\{x_i\}} \gamma$ if there exist a pair of integers j, k, such that

$$x_j < x_k, \quad x_{\gamma(j)} > x_{\gamma(k)}, \quad \gamma'(j) = \gamma(k), \quad \gamma'(k) = \gamma(j), \quad \gamma'(i) = \gamma(i),\ i \neq j,\ i \neq k.$$

2. $\gamma' \prec_{\{x_i\}} \gamma$ if there exists γ'' such that $\gamma' \prec_{\{x_i\}} \gamma''$ and $\gamma'' \prec_{\{x_i\}} \gamma$.

It is easy to see that $\prec_{\{x_i\}}$ is a partial order on Γ. Note that $\prec_{\{x_i\}}$ and $\prec_{\{x'_i\}}$ define the same order provided

$$\forall i,j : x_i < x_j \quad \text{iff} \quad x'_i < x'_j.$$

Owing to Corollary 3.3, we get

Corollary 3.5 *Let $y_1 \leq y_2 \leq \cdots \leq y_n$ be real numbers. If $\gamma' \prec_{\{x_i\}} \gamma$, then*

$$(y_{\gamma'} - x) \prec (y_\gamma - x). \tag{3.10}$$

4 Optimal Properties of FCFS Policies

In this section, we consider the case where no information about reference times is available. This occurs when there are no reference times associated with jobs, or when the reference times are not known a priori. We take the number of jobs in the system, the throughput, and the response times of tasks and of jobs as the performance criteria.

4.1 Minimizing the Number of Jobs and the Throughput

To begin with, we show some optimal properties of FCFS policies with respect to the number of jobs and to the throughput.

Proposition 4.1 *If assumption A holds, then non-preemptive FCFS policies minimize the number of jobs over the class of non-preemptive policies, with respect to the strong ordering:*

$$\forall \pi \in \Pi, \quad \exists \xi \in \Xi : \quad N_t(\xi) \leq_{st} N_t(\pi), \qquad \forall t > 0. \tag{4.1}$$

Proof. Suppose $m \geq 1$ jobs have arrived by time $t \geq 0$: $a_1 < a_2 < \cdots < a_m \leq t < a_{m+1} < \cdots$. Let $\{b_{i,n}(\pi)\}_{n=1}^m$ and $\{c_{i,n}(\pi)\}_{n=1}^m$, $1 \leq i \leq |V|$, be the sequences defined for the policy π. Let $I_\pi(i,n)$ denote the job index of the task which is the n-th to leave the system among tasks $T_{i,1}, T_{i,2}, \cdots$, according to policy π. By definition, $\{b_{i,I_\pi(i,n)}(\pi)\}_{n=1}^m$ and $\{c_{i,I_\pi(i,n)}(\pi)\}_{n=1}^m$, $1 \leq i \leq |V|$, are non-decreasing sequences.

Now let ψ be a new non-preemptive scheduling policy which starts attending tasks according to the sequence $\{b_{i,n}(\psi)\}_{n=1}^m$ identical to $\{b_{i,I_\pi(i,n)}(\pi)\}_{n=1}^m$, $1 \leq i \leq |V|$. Here ψ may be idling even if π is a non-idling policy. Policy ψ operates on the service time sequences $\{\hat{s}_{i,n}\}_{n=0}^m$, $1 \leq i \leq |V|$, which are coupled with the initial ones in such a way that $\hat{s}_{i,n} = s_{i,I_\pi(i,n)}$, $1 \leq i \leq |V|$, $1 \leq n \leq m$. Such an interchange has no impact on the law of the departure process in view of the assumption that the task service times are i.i.d. Hence, $\{c_{i,n}(\psi)\}_{n=1}^m$, is identical to $\{c_{i,I_\pi(i,n)}(\pi)\}_{n=1}^m$. Observe that ψ is FIFO (First In First Out) at task level.

We show that ψ is feasible. Relation (2.1) follows from the feasibility of π. In fact

$$\max_{1 \leq m \leq n} I_\pi(i,m) \geq n.$$

Therefore,

$$b_{i,n}(\psi) = \max_{1 \leq m \leq n} b_{i,m}(\psi) = \max_{1 \leq m \leq n} b_{i,I_\pi(i,m)}(\pi) \geq \max_{1 \leq m \leq n} a_{I_\pi(i,m)} \geq a_n.$$

Property (2.2) is shown ab absurdo. Suppose (2.2) is false, i.e., there exist some i, n and j, l, such that

$$\mathcal{M}(i) = \mathcal{M}(j), \quad \text{and} \quad b_{i,n}(\psi) < b_{j,l}(\psi) < c_{i,n}(\psi). \tag{4.2}$$

Rewriting the above relation in terms of the service patterns of π yields

$$\mathcal{M}(i) = \mathcal{M}(j), \quad \text{and} \quad b_{i,I_\pi(i,n)}(\pi) < b_{j,I_\pi(j,l)}(\pi) < c_{i,I_\pi(i,n)}(\pi) \tag{4.3}$$

which contradicts the feasibility of π and the fact that it is non-preemptive. Thus property (2.2) is proved.

Relation (2.3) is equivalent to the relation

$$b_{i,n}(\psi) \geq c_{j,n}(\psi) \qquad \forall (j,i) \in E \tag{4.4}$$

or

$$b_{i,I_\pi(i,n)}(\pi) \geq c_{j,I_\pi(j,n)}(\pi) \qquad \forall (j,i) \in E. \tag{4.5}$$

This last relation is a direct consequence of the relation

$$\nu_{i,n} \stackrel{\text{def}}{=} \sum_{l=1}^{m} \mathbf{1}_{\{c_{j,l}(\pi) \leq b_{i,I_\pi(i,n)}(\pi)\}} \geq n \qquad \forall (j,i) \in E, \forall n: \ 1 \leq n \leq m. \tag{4.6}$$

In words, for all pair of indices (j,i) such that j is a predecessor of i in G, for all $1 \leq n \leq m$, there are at least n tasks of type T_j which have left the system before task $T_{i,I_\pi(i,n)}$ (which is the n-th that is processed in the sequence of tasks $T_{i,1}, T_{i,2}, \cdots, T_{i,m}$) begins processing.

It follows from (4.6) that for all pair of indices $(j,i) \in E$, and for all $1 \leq n \leq m$,

$$c_{j,n}(\psi) = c_{j,I_\pi(j,n)}(\pi) \leq c_{j,I_\pi(j,\nu_{i,n})}(\pi) \leq b_{i,I_\pi(i,n)}(\pi) = b_{i,n}(\psi).$$

Now we can construct an *improved* FCFS policy ξ from ψ by processing tasks as early as possible. More precisely, assume that the service times for ξ are the same as ψ. We can recursively define the sequences of attention times and completion times of ξ, $\{b_{i,n}(\xi)\}_{n=1}^{m}$ and $\{c_{i,n}(\xi)\}_{n=1}^{m}$, $1 \leq i \leq |V|$, in the following way:

$$b_{1,1}(\xi) = b_{1,1}(\psi), \tag{4.7}$$

$$b_{i,n}(\xi) = \max \{ a_n, \max_{\{j \in p(i)\}} c_{j,n}(\xi), \max_{\{j | \mathcal{M}(j) = \mathcal{M}(i)\}} c_{j,n-1}(\xi), \max_{\{j | \mathcal{M}(j) = \mathcal{M}(i), b_{j,n}(\psi) < b_{i,n}(\psi)\}} c_{j,n}(\xi) \}, \tag{4.8}$$

$$c_{i,n}(\xi) = b_{i,n}(\xi) + c_{i,n}(\psi) - b_{i,n}(\psi), \tag{4.9}$$

where by convention, $c_{i,-1} = -\infty$.

The feasibility of ξ is easily proved. Indeed, properties (2.1) and (2.3) for ξ come immediately from equation (4.8). Property (2.2) is also guaranteed by (4.8), as all the services of ξ for the tasks assigned to a same machine are ordered in the same way as in ψ.

Using (4.7)-(4.9), one can now check by induction on i and n that

$$c_{i,n}(\xi) \leq c_{i,n}(\psi) = c_{i,I_\pi(i,n)}(\pi), \qquad 1 \leq i \leq |V|, \quad 1 \leq n \leq m. \tag{4.10}$$

Consequently, we have found a new FCFS policy ξ which, when applied to a specific realization of the service times, satisfies the ordering relation

$$N'_t(\xi) \leq N_t(\pi)$$

where the prime indicates that ξ operates here on this specific realization of the sequence of service times. Hence, for all $f : I\!N \rightarrow I\!R$ non decreasing,

$$f(N'_t(\xi)) \leq f(N_t(\pi))$$

so that

$$E[f(N'_t(\xi))] \leq E[f(N_t(\pi))]$$

whenever the expectations exist. Since the two realizations of the sequence of service times are equivalent in law, we have

$$E[f(N_t(\xi))] = E[f(N'_t(\xi))]$$

so that

$$E[f(N_t(\xi))] \leq E[f(N_t(\pi))]$$

which completes the proof. ∎

Note that π in the above proposition may be an idling policy. However, if the precedence relation on the set of the tasks assigned to the same machine defines a total order (which is analyzed in [1]), the resulting FCFS policy ξ can be shown to be non-idling.

The following result follows immediately from Proposition 4.1.

Corollary 4.2 *If assumption A holds, then non-preemptive FCFS stochastically maximizes the throughput over the class of non-preemptive policies:*

$$\forall \pi \in \Pi, \quad \exists \xi \in \Xi : \qquad \mathcal{T}_t(\xi) \geq_{st} \mathcal{T}_t(\pi), \qquad \forall t > 0. \tag{4.11}$$

4.2 Minimizing the Response Times

As stated in the above subsection, FCFS policies are optimal in terms of the number of jobs and the throughput, with respect to the strong stochastic ordering $\leq_{st}$. Using the idea of the proof and the results of Section 3, we obtain the following additional relation concerning Response Times:

Proposition 4.3 *Let N be a given non-negative integer, and*

$$\vec{R}^i(\pi) = (R_1^i(\pi), R_2^i(\pi), \cdots, R_N^i(\pi)).$$

If assumption A holds, then the relations

$$\forall \pi \in \Pi,\ \exists \xi \in \Xi: \qquad \vec{R}^i(\xi) \leq_{scx} \vec{R}^i(\pi) \tag{4.12}$$

hold for all $i = 1, \cdots, |V|$.

Proof. Let $\{b_{i,n}(\pi)\}_{n=1}^{\infty}$ and $\{c_{i,n}(\pi)\}_{n=1}^{\infty}$, $1 \leq i \leq |V|$, be the sequences of service attention and completion times of the tasks associated with π.

As in the proof of Proposition 4.1, by interchanging service times, one can obtain a (possibly idling) feasible scheduling ψ, which is FCFS. More precisely, the first policy ψ that is defined in the proof of Proposition 4.1 is characterized by increasing sequences $\{b_{i,n}(\psi)\}_{n=1}^{\infty}$ and $\{c_{i,n}(\psi)\}_{n=1}^{\infty}$, $1 \leq i \leq |V|$, which are mere permutations of the initial sequences $\{b_{i,n}(\pi)\}_{n=1}^{\infty}$ and $\{c_{i,n}(\pi)\}_{n=1}^{\infty}$: For all $1 \leq i \leq |V|$, $n \geq 0$, there exists $m \geq 1$ such that $b_{i,n}(\psi) = b_{i,m}(\pi)$ and $c_{i,n}(\psi) = c_{i,m}(\pi)$.

Assumption A implies that the interchanges of the service orders and service times do not change the law of the departure sequence. Therefore Corollary 3.4 entails that

$$f(\vec{R}^{i\prime}(\psi)) \leq f(\vec{R}^i(\pi)) \tag{4.13}$$

holds for all Schur convex function $f: IR^N \to IR$, where the prime refers to the interchange of service times as earlier. Owing to the i.i.d. assumption, we get then that the relation

$$E[f(\vec{R}^i(\psi))] \leq E[f(\vec{R}^i(\pi))] \tag{4.14}$$

holds for all Schur convex function $f: IR^N \to IR$ such that the expectations exist. ∎

Concerning the response time of jobs, we have the following similar results:

Proposition 4.4 *Let N be a given non-negative integer, and*

$$\vec{R}(\pi) = (R_1(\pi), R_2(\pi), \cdots, R_N(\pi)).$$

Assume A holds, then

$$\forall \pi \in \Pi,\ \exists \xi \in \Xi: \qquad \vec{R}(\xi) \leq_{scx} \vec{R}(\pi). \tag{4.15}$$

Proof. Consider a task graph $G' = (V', E')$ which is obtained by adding a fictitious task $T_{|V|+1}$ into the task set V. This task requires no service time and it is the successor of all the other tasks in V:

$$E' = E \bigcup \{(i, |V| + 1) | 1 \leq i \leq |V|\}$$

The fictitious task will be assigned to a fictitious machine whose only tasks are $T_{|V|+1,n}$, $n \geq 1$. Note that G' is still acyclic. Moreover, the introduction of such fictitious tasks does not affect the behavior of the initial processing system (see Baccelli and Liu [2]). Therefore, for all $\pi \in \Pi$,

$$R_n(\pi) = \max_{1 \leq i \leq |V|} R_n^i(\pi) = R_n^{|V|+1}(\pi).$$

The assertion of the Proposition then follows from Proposition 4.3. ∎

5 Shortest Reference First and Longest Reference First Policies

We consider now the case when jobs are constrained by some reference times, d_n. We are interested in the job lag time, which is defined as the difference between the job completion time and its reference time : $c_n - d_n$. We assume that these reference times are mutually different.

5.1 Some Facts and Counterexamples

Two classes of extremal policies are easily defined. The first class consists in scheduling first the tasks (and hence the jobs) with the shortest reference time. Such policies are called Shortest Reference First (SRF) policies. The second class consists in scheduling the tasks with the Longest Reference Time, and it is hence called the class of Longest Reference First (LRF) policies.

In Towsley and Baccelli [9], it is shown that in a tandem network composed of $G/GI/1$ queues, for all convex functions f : $I\!R \to I\!R$,

$$\sum_{i=1}^{N} E[f(L_i(SRF))] \leq \sum_{i=1}^{N} E[f(L_i(\pi))] \leq \sum_{i=1}^{N} E[f(L_i(LRF))],$$

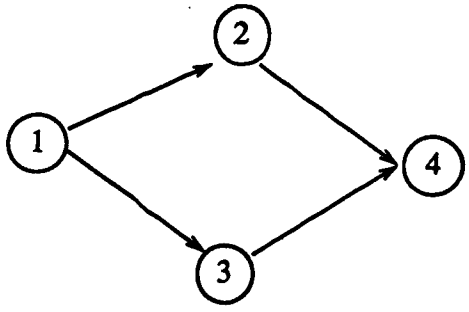


Figure 1: A counterexample of the optimality of SRF

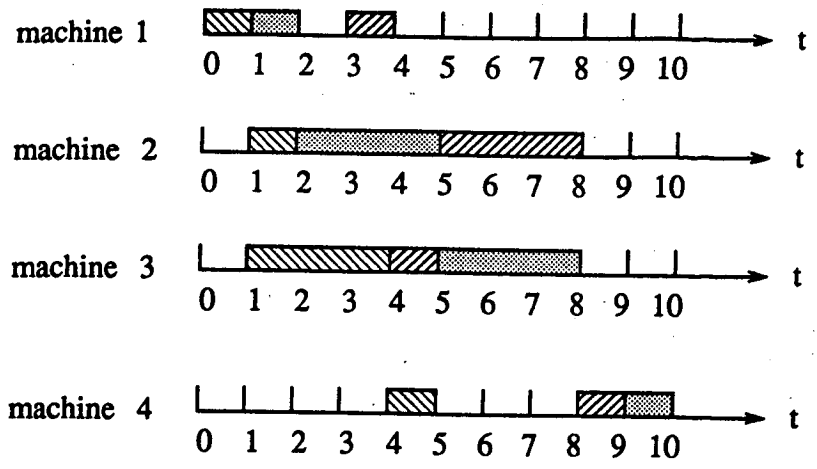
where $L_i(\pi)$ (resp. $L_i(SRF)$, $L_i(LRF)$) denotes the lag time of the i-th customer (arrived to the system) obtained with policy π (resp. SRF, LRF). This relation yields the following convex ordering relation on the stationary lag times, provided the system satisfies some natural ergodicity condition

$$L(SRF) \leq_{cx} L(\pi) \leq_{cx} L(LRF).$$

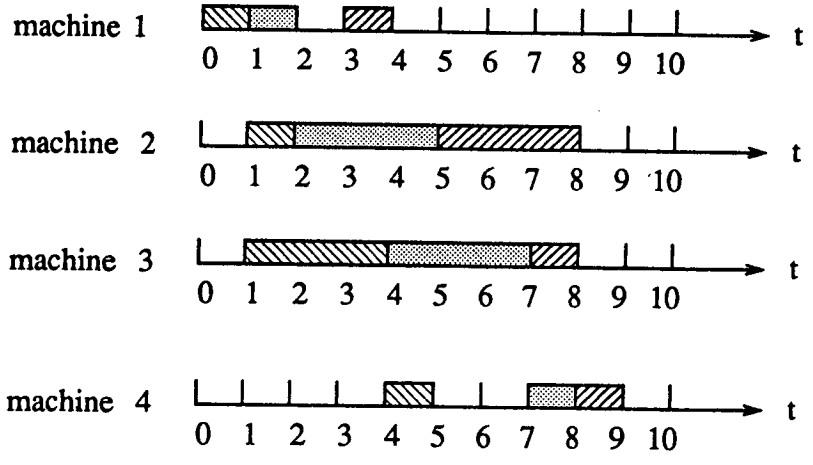
It is therefore natural to ask whether such results extend to the parallel processing systems that we analyze. Unfortunately, the results of the previous sections do not generalize unless some extra assumptions are made. One can easily construct simple counterexamples showing that these results do not hold in general when reference times are taken into account. Consider the following one.

Let a job be structured as a series-parallel graph with a fork task T_1, followed by two parallel tasks T_2, T_3, and finishing with a join task T_4 (cf. Figure 1). Suppose there are four machines, one for each task. Assume that the processing times of T_1 and T_4 are equal to 1, and that those of T_2 and T_3 are equal to 1 or 3 with probability 1/2. The relative reference times of the jobs are equal to 5 or 8 with probability 1/2. Consider the sample path with the characteristics of the jobs given in Table 1(a). Figure 2 provides the Gantt chart [4] of the SRF and the FCFS policies. One observes immediately that for this particular sample path, FCFS is better than SRF (cf. Table 1(b)).

The interested reader can check that with the job arrival times described by Table 1(a), the *average* lag time obtained by FCFS is also smaller than that by SRF (the difference is $-7/64$). Counterexamples showing that LRF is not extremal can also be constructed in a similar way. The reason for which SRF and LRF policies fail to satisfy the earlier extremal pathwise properties is that the orders in which tasks $T_1, T_2, \cdots$ are attended may be different, which results in an increased synchronization overhead.



(a)



(b)

Figure 2: (a) Gantt chart of SRF; (b) Gantt chart of $FCFS$.

n	a_n	d_n	$\sigma_{1,n}$	$\sigma_{2,n}$	$\sigma_{3,n}$	$\sigma_{4,n}$
1	0	5	1	1	3	1
2	1	9	1	3	3	1
3	3	8	1	3	1	1

(a)

n	$c_n(SRF)$	$L_n(SRF)$	$c_n(FCFS)$	$L_n(FCFS)$
1	5	0	5	0
2	10	1	8	-1
3	9	1	9	1

(b)

Table 1: Counterexample of SRF optimality. (a) Job timings, (b) Lag times.

The following question naturally arises: What further constraints should be imposed on the class of scheduling policies in order to make SRF and LRF policies extremal again. The answer to the question lies in the notion of Service Order Preserving policies that is introduced below.

5.2 Service Order Preserving Policies

We define two types of Service Preserving Policies: Global Order Preserving (GOP) and Local Order Preserving (LOP) policies. Let $I_\pi(k,n)$, $n = 1,2,\cdots$ denote the identity of the job associated with the completion of the n-th instance of task T_k.

Definition 5.1 *A policy π is GOP if $I_\pi(j,n) = I_\pi(k,n)$, $n = 1,2,\cdots$, for every pair of tasks T_j, T_k.*

An example of a global order preserving policy is the one that schedules all instances of the same task in the order of job arrival as in FCFS policies.

Before defining LOP policies, we define the *series decomposition* of a task graph.

An acyclic graph G has a series representation, $\mathcal{G}$, denoted by

$$\mathcal{G} = ((G^1, A^1, B^1), (G^2, A^2, B^2), \cdots, (G^{g-1}, A^{g-1}, B^{g-1}), (G^g, A^g, B^g))$$

where $g = |\mathcal{G}|$ is a positive integer, $G^i = (V^i, E^i)$, $i = 1, 2, \cdots, g$, are disjoint subgraphs of G, and

$$
\begin{aligned}
V &= \bigcup_{i=1}^{g} V^i, \\
E &= \left(\bigcup_{i=1}^{g} E^i\right) \bigcup \{(u,v) : \forall u \in B^k, \forall v \in A^{k+1}, k = 1, \cdots, g-1\}, \\
A^i &\subseteq V^i, \quad i = 1, \cdots, g, \\
B^i &\subseteq V^i, \quad i = 1, \cdots, g, \\
p(A^1) &= \emptyset, \\
s(B^g) &= \emptyset, \\
s(j) &= A^{i+1}, \quad \forall j \in B^i, \quad i = 1, \cdots, g-1.
\end{aligned}
$$

In words, G is decomposed into a series subgraphs $G^1, \cdots, G^g$ in such a way that the vertices in V^i are the (not necessarily immediate) predecessors of every vertex in V^{i+1}, $i = 1, 2, \cdots, g-1$. The set of vertices A^i (resp. B^i) represents the vertices without predecessors (resp. successors) in G^i. In addition, all vertices of A^{k+1} must admit all vertices of B^k as predecessors for all $k = 1, \cdots, g-1$, which requires a fairly strong sort of synchronization points between the subgraphs.

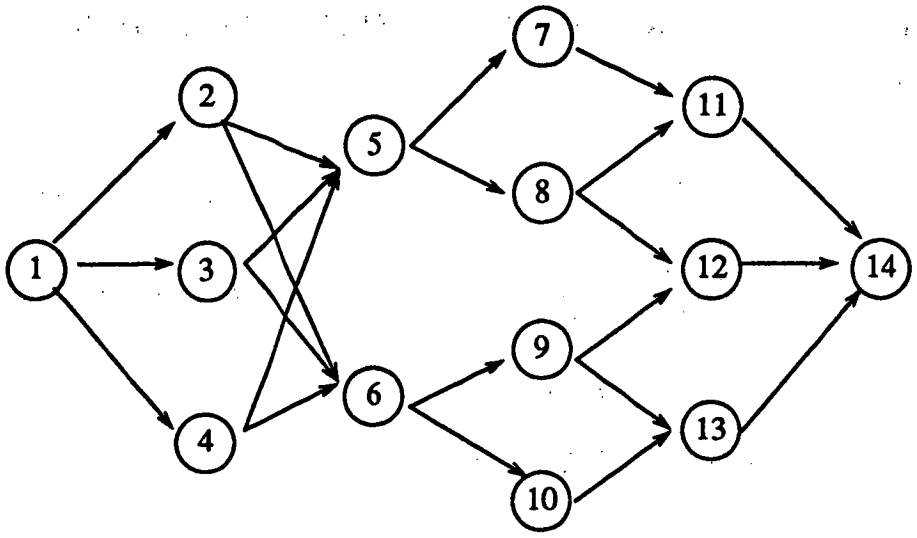
The simplest example arises in the series graph. A series-parallel graph is also easily seen to be series decomposable, provided it is the concatenation of at least two series parallel graphs. Note that it is not always possible to decompose an acyclic graph into two or more subgraphs.

The series representation, $\mathcal{G}$, of an acyclic graph G is said to be the *minimal series representation* of G if there is no other series representation $\mathcal{G}'$ of G such that $|\mathcal{G}| < |\mathcal{G}'|$. Series decompositions exhibit the following property.

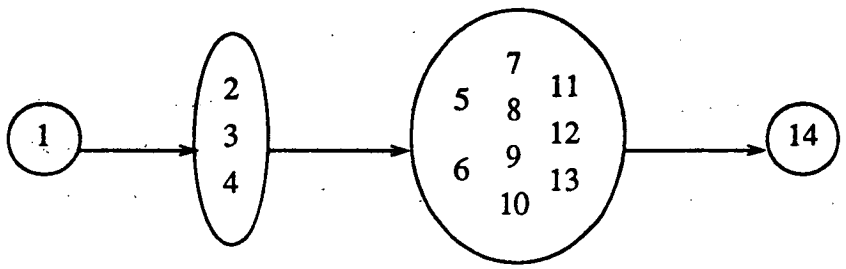
Property 5.2 *There is a unique minimal series representation of an acyclic graph.*

An acyclic task graph with its minimal series decomposition is illustrated in Figure 3.

A simple algorithm constructing the minimal series representation $\mathcal{G}$ of G is provided in the Appendix.



(a)



(b)

Figure 3: Sample task graph and minimal series decomposition.

We are now in a position to introduce the notion of Local Order Preserving (LOP) scheduling. Consider the task graph G with its minimal series representation $\mathcal{G}$. Let $I_\pi(k,m,n)$, $k \in V^m$, $1 \leq m \leq g$, $n = 1,2,\cdots$ denote the identity of the job associated with the completion of the n-th instance of task T_k in subgraph G^m.

Definition 5.3 *A policy π is a LOP policy if $I_\pi(k_1,m,n) = I_\pi(k_2,m,n)$, $n = 1,2,\cdots$ for every pair of tasks $k_1, k_2 \in V^m$ and for all subgraphs G^m, $m = 1,\cdots,g$.*

Observe that any GOP policy is LOP.

Remarks:

1. In order to check that a policy is LOP or GOP, it is convenient to associate a *virtual queue* with each task in the task graph. The customers of the virtual queue associated with T_j are the instances of the task, $T_{j,n}$, $n \geq 1$. If T_j has no predecessor in G, then $T_{j,n}$ arrives to the virtual queue at time a_n, otherwise, it arrives to the queue at the epoch when all of its predecessors have left their corresponding virtual queues. Task $T_{j,n}$ is attended at time $b_{j,n}(\pi)$ and leaves the virtual queue at time $c_{j,n}(\pi)$, when policy π is used. The virtual queue can therefore be viewed as a possibly idling single server queue.

2. Policy π is LOP if and only if for all indices i and j such that T_i and T_j belong to the same subgraph, tasks $T_{i,1}$, $T_{i,2},\cdots$ and $T_{j,1}$, $T_{j,2},\cdots$ are served in the same order in the virtual queues associated with i and j respectively. However, the tasks belonging to the same job need not be served in the same position if they belong to different subgraphs. If this additional property is satisfied, policy π is GOP.

3. Note that the virtual queues associated with the tasks can also be used to implement the LOP or GOP policies. Indeed, if an LOP policy π is to be implemented, every time π attends a task, it scans the virtual queues associated with the tasks of the same subgraph in order to select which of the instances of the task is to be attended. In order to get the LOP property, this selection should operate as follows. For any time t, let $M_t(j,\pi)$ be the number of customers already served or currently being served by time t in the virtual queue associated with task T_j. Let $j(i)$ be the index of the virtual queue where the number of customers attended or being attended by time t is the largest among the virtual queues associated with the tasks in subgraph G^i:

$$M_t(j(i),\pi) = \max_{j \in G^i} M_t(j,\pi).$$

If $M_t(j,\pi) = M_t(j(i),\pi)$, π is free to choose whichever instances of task T_j are present in the queue. If $M_t(j,\pi) < M_t(j(i),\pi)$, π has to select task T_{j,m^*} for service, where m^* is the index of the task that is the m-th to be attended by the virtual queue associated with $T_{j(i)}$, provided this task is present in the virtual queue. Otherwise, the virtual queue remains idle until this task arrives.

4. In order to visualize the behavior of a LOP policy it is also useful to associate another type of virtual queues with the subgraphs of $\mathcal{G}$. The customers of the virtual queue associated with G^i are the instances of the subgraph, G_n^i. Customer G_n^1 arrives to the virtual queue at time a_n. For all $2 \leq i \leq g$, G_n^i arrives to the queue when G_n^{i-1} leaves the virtual queue associated with the subgraph G^{i-1}. Customer G_n^i is attended by its virtual queue as soon as the first task of G_n^i is attended by the virtual queue associated with the task. G_n^i remains in this virtual queue until the moment that all the tasks of G_n^i have left their corresponding virtual queues. Several customers can be served simultaneously, so that the virtual queues associated with subgraphs may be viewed as a multiple server queue.

5. As an example, consider the serial task graph in Figure 4(a). If each task is allocated to a separate machine, then the virtual queueing systems associated with the tasks and the subgraphs coincide, and correspond to the tandem queueing network in Figure 4(b).

Let Θ denote the class of non-preemptive LOP policies. As mentioned earlier, FCFS policies are GOP and therefore $\Xi \subset \Theta$. Other sub-classes of LOP policies of interest include:

- Ψ - the class of non-preemptive LOP policies that do not use reference times in scheduling customers.

- Σ - the class of non-preemptive LOP SRF policies. An LOP policy belongs to this class if, in the virtual queueing system associated with the subgraphs, for every virtual queue, the customer with the shortest reference time is attended first.

- Λ - the class of non-preemptive LOP LRF policies. An LOP policy belongs to this class if, in the virtual queueing system associated with the subgraphs, for every virtual queue the customer with the longest reference time is attended first.

- Υ - The class of non-preemptive LOP policies where all the virtual queues associated with the subgraphs attend the latest arrived customer.

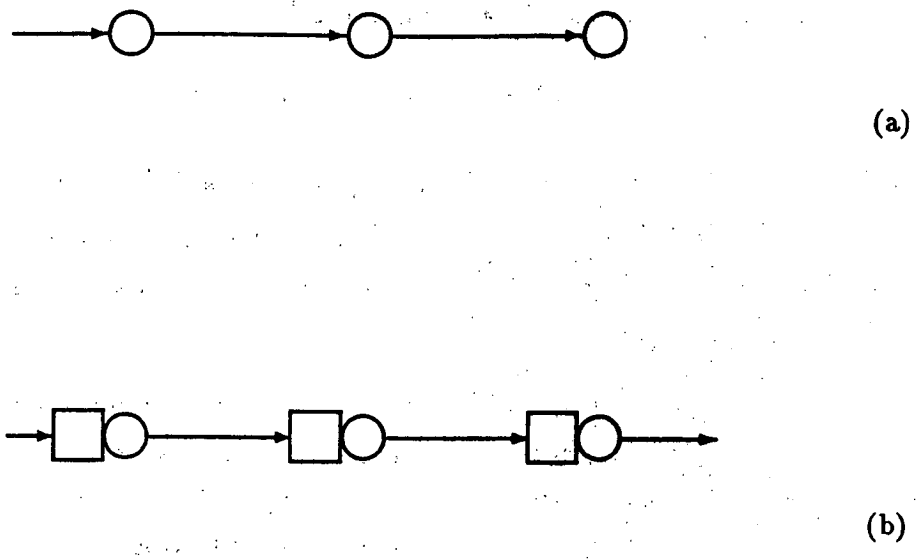


Figure 4: Example of virtual queueing network representation of LOP policy.

Θ', Ψ', Ξ', Υ', Σ', Λ' will denote the preemptive resume versions of the classes defined above.

One may also define similar classes of global order preserving policies. Note that these policies are possibly idling.

5.3 Optimal Properties of LOP Policies

The first interesting property of LOP scheduling is that LOP policies provide optimal throughput. We show this result by first proving that policies in Ξ and those in Θ provide the same performance with respect to the number of jobs and to throughput, and by using then the results of Section 4.

Proposition 5.4 *If assumption A holds, then*

$$\forall \theta \in \Theta, \quad \exists \xi \in \Xi: \quad N_t(\theta) =_{st} N_t(\xi), \qquad \forall t > 0 \tag{5.1}$$

$$\forall \theta \in \Theta, \quad \exists \xi \in \Xi: \quad \mathcal{T}_t(\theta) =_{st} \mathcal{T}_t(\xi), \qquad \forall t > 0 \tag{5.2}$$

Proof. Let $G^1, \cdots, G^g$ be the series decomposition of graph G, with $G^1 \rightarrow \cdots \rightarrow G^g$. Let $J_\theta(h,n)$ denote the job index of the subgraph which is the n-th to be attended among

the set $G_1^h, G_2^h, \cdots$, $1 \leq h \leq g$.

Define ξ to be the FCFS policy obtained by permuting the service orders of θ in such a way that FCFS is applied to all the subgraphs $G^1, \cdots, G^g$:

$$J_\xi(h,n) = n \qquad h = 1, \cdots, g, \quad n \geq 1.$$

Here, a permutation of the service order of subgraph G_m^h and G_n^h indicates that the order of service of the tasks $T_{i,m} \in G_m^h$ and $T_{i,n} \in G_n^h$ are permuted. Moreover, we consider the version where the service times are also interchanged. As shown in Proposition 4.1, ξ is a feasible policy.

It is not difficult to see that ξ and θ generate the same subgraph departure sequences from the virtual queues associated with the subgraphs $G^1, \cdots, G^g$. Since the job departure process coincides with the departure process of subgraphs G^g, it follows immediately that ξ and θ generate the same job departure sequences as well. Note that under assumption A, the interchanges of service times which allow us to obtain ξ do not change the law of the departure sequence. Therefore, the departure sequences of ξ and θ have the same distribution. ∎

5.4 Extremal Properties of SRF and LRF Policies

This section focuses on the extremal properties of SRF and LRF within the class of LOP policies.

For $n = 1, 2, \cdots$, let $b_n^i(\pi)$ (resp. $c_n^i(\pi)$) denote the attention (resp. completion) time of subgraph G_n^i in policy π, i.e., the minimum (resp. maximum) of the attention (resp. completion) times of the tasks of G_n^i in policy π, where the subgraphs $G_n^1, G_n^2, \cdots, G_n^g$ are minimal elements of the series decomposition of G_n. We set $c_n^0 = a_n$.

Proposition 5.5 *Let N be a given non-negative integer, and*

$$\vec{L}(\theta) = (L_1(\theta), L_2(\theta), \cdots, L_N(\theta)).$$

If assumption A holds, then

$$\forall \theta \in \Theta, \ \exists \sigma \in \Sigma, \ \lambda \in \Lambda: \qquad \vec{L}(\sigma) \leq_{scx} \vec{L}(\theta) \leq_{scx} \vec{L}(\lambda). \tag{5.3}$$

Proof. We prove the first inequality. The second one is proven in a similar manner.

Let $I_\theta(i,n)$ denote the identity of the n-th subgraph G^i to be served in the corresponding virtual queue, according to the LOP policy θ. Let $\vec{H}_i(\theta)$ denote the lag time vector seen by subgraphs $G^i_1, G^i_2, \cdots, G^i_N$:

$$\vec{H}_i(\theta) = (c^i_1(\theta) - d_1, c^i_2(\theta) - d_2, \cdots, c^i_N(\theta) - d_N).$$

It is clear that $\vec{H}_g(\theta) = \vec{L}(\theta)$.

Note that a LOP policy η follows the SRF rule in processing the subgraphs G^i if and only if

$$\forall m,n:\ \min(b^i_m(\eta), b^i_n(\eta)) \geq \max(c^{i-1}_m(\eta), c^{i-1}_n(\eta)),\ d_m < d_n \quad \Rightarrow \quad b^i_m(\eta) \leq b^i_n(\eta). \tag{5.4}$$

In words, the virtual queue associated with G^i applies the SRF policy if and only if, at any decision point, it attends the customer with the shortest reference time among the customers that are present in the queue.

Based on the LOP policy θ, we define some new policies $\eta_i \in \Theta$, $i = 0, \cdots, g$, with $\eta_0 = \theta$ and such that η_i satisfies the SRF rule for all subgraphs $G^1, \cdots, G^i$, $i = 1, \cdots, g$.

Policy η_i is defined as follows: Suppose for some $i \geq 1$, policy η_{i-1} is already defined and attends the subgraphs $G^1, \cdots, G^{i-1}$ according to the SRF rule. If η_{i-1} follows the SRF rule for subgraph G^i, then $\eta_i = \eta_{i-1}$. Otherwise, for any pair of integer m, n, such that relation (5.4) is not satisfied for G^i, define a new policy η identical to η_{i-1} except for G^i_m and G^i_n, where

$$I_\eta(l,j) = \begin{cases} I_{\eta_{i-1}}(l,j), & j = 1, N,\ 1 \leq l \leq i-1, \\ I_{\eta_{i-1}}(l,j), & j \neq m,\ j \neq n,\ i \leq l \leq g, \\ \min\{I_{\eta_{i-1}}(l,n), I_{\eta_{i-1}}(l,m)\}, & j = m,\ i \leq l \leq g, \\ \max\{I_{\eta_{i-1}}(l,n), I_{\eta_{i-1}}(l,m)\}, & j = n,\ i \leq l \leq g. \end{cases}$$

Therefore, η is the same as η_{i-1} concerning subgraphs $G^1, \cdots, G^{i-1}$. For those tasks in subgraphs $G^i, \cdots, G^N$, the service order for tasks $T_{k,n}$ and $T_{k,m}$ are permuted if $T_{k,m}$ was served prior to $T_{k,n}$. As above, we consider the version of the policy where the service times of the tasks $T_{k,m}$ and $T_{k,n}$ are interchanged whenever the order of service is interchanged. This version of the policy is equivalent in law to the version without interchange for the same reason as above.

The feasibility of η can easily be checked from the fact that

$$b_n^i(\eta) \geq c_n^{i-1}(\eta), \quad 1 \leq i \leq g.$$

After a finite number of such permutations of attention times, we obtain a (uniquely defined) policy η_i which follows the SRF rule with respect to subgraphs $G^1, \cdots, G^i$ and that is still LOP by construction.

Observe that from any feasible policy η obtained from θ by permuting the attention times in θ, and by interchanging the service times accordingly, this procedure yields the same SRF policy η_g, provided the reference times are mutually different. This property can be shown by induction on $i = 1, 2, \cdots, g$.

For any policy π, define

$$\vec{I}_\pi(i) = (I_\pi(i,1), I_\pi(i,2), \cdots, I_\pi(i,N)).$$

It should be clear from the construction of η_i that

$$\vec{I'}_{\eta_i}(j) \prec_{\{d_n\}} \vec{I}_{\eta_{i-1}}(j), \qquad i = 1, 2, \cdots, g; j = 1, \cdots, g. \tag{5.5}$$

where the prime refers to the fact that the sequence of service times is not the same as the initial one. Consequently

$$\vec{I'}_{\eta_g}(g) \prec_{\{d_n\}} \vec{I}_\theta(g). \tag{5.6}$$

As a consequence of (5.6) and Corollary 3.5, we obtain

$$\vec{H'}_i(\eta_g) = \vec{H'}_i(\eta_i) \prec \vec{H}_i(\theta), \qquad i = 1, 2, \cdots, g, \tag{5.7}$$

which implies

$$\vec{L'}(\eta_g) = \vec{H'}_g(\eta_g) \prec \vec{H}_g(\theta) = \vec{L}(\theta). \tag{5.8}$$

Under assumption A, the interchanges of the service orders and service times do not change the law of the departure sequence, so that this last relation implies that

$$E[f(\vec{L}(\eta_g))] \leq E[f(\vec{L}(\theta))] \tag{5.9}$$

holds for all Schur convex function $f : I\!R^N \to I\!R$ such that the expectations exist. The assertion of the proposition is thus proved. ∎

In words, SRF policies minimize (resp. maximize) Schur convex and hence convex symmetric (resp. Schur concave and concave symmetric) functions of (transient) lag times and minimize (resp. maximize) convex functions of stationary lag times, whereas LRF policies maximize (resp. minimize) Schur convex and convex symmetric (resp. Schur concave and concave symmetric) functions of (transient) lag times and maximize (resp. minimize) concave functions of stationary lag times.

5.5 Extremal Properties of FCFS and LCFS Policies

As an application of the results of the previous subsection, we consider again the case where no information about reference times is available, so that the SRF and LRF policies analyzed above are not implementable.

When setting the reference times to the arrival times of the jobs: $d_n = a_n$, $n = 1, 2, \cdots$, we obtain that the lag time of a job equals the response time of the job, and that the SRF (resp. LRF) policies are equivalent to FCFS (resp. LCFS with local service order preserving) policies: $\Sigma = \Xi$ and $\Lambda = \Upsilon$ (In fact, these last two relations hold as soon as $d_1 < d_2 < d_3 < \cdots$). Applying the result in the previous subsection immediately yields

Proposition 5.6 *Let N be a given non-negative integer, and*

$$\vec{R}(\pi) = (R_1(\pi), R_2(\pi), \cdots, R_N(\pi)).$$

Assume A holds, then

$$\forall \theta \in \Theta,\ \exists \xi \in \Xi,\ \upsilon \in \Upsilon: \qquad \vec{R}(\xi) \leq_{scx} \vec{R}(\theta) \leq_{scx} \vec{R}(\upsilon) \tag{5.10}$$

In the case that the reference times are not identical to the arrival times and are unknown a priori, we have the following convex symmetric ordering result.

Corollary 5.7 *Let N be a given non-negative integer, and*

$$\vec{L}(\pi) = (L_1(\pi), L_2(\pi), \cdots, L_N(\pi)).$$

Assume A holds and that the relative reference times form an i.i.d. sequence of RVs. Then

$$\forall \psi \in \Psi,\ \exists \xi \in \Xi,\ \upsilon \in \Upsilon: \qquad \vec{L}(\xi) \leq_{cxs} \vec{L}(\psi) \leq_{cxs} \vec{L}(\upsilon) \tag{5.11}$$

Proof. We have

$$\vec{L}(\psi) = (R_1(\psi) - r_1, R_2(\psi) - r_2, \cdots, R_N(\psi) - r_N).$$

where $r_1, \cdots, r_N$ are i.i.d random relative reference times being mutually independent of the response time variables. Let $f : I\!R^N \to I\!R$ be a *cxs* function and $g : I\!R^N \to I\!R$ be defined by

$$g(x_1, \cdots, x_N) = \int_{-\infty}^{\infty} \cdots \int_{-\infty}^{\infty} f(x_1 - u_1, \cdots, x_N - u_N)\rho(du_1) \cdots \rho(du_N)$$

where ρ denotes the distribution function of the relative reference times. It is easily checked that g is also *cxs*. This together with the fact that

$$E[f(L_1(\psi), \cdots, L_N(\psi))] = E[g(R_1(\psi), \cdots, R_N(\psi))]$$

allow one to conclude the proof in view of Proposition 5.6.

6 Extremal Properties in Steady State

In this section, we discuss the extremal properties of the FCFS, LCFS, LOP, SRF and LRF scheduling policies in steady state, provided it exists.

6.1 FCFS in Steady State

From the results of Section 4, we get

Corollary 6.1 *If assumption A holds, and if the policies π and ξ defined in Proposition 4.1 are such that the random variables $N_t(\pi)$ and $N_t(\xi)$ (resp. $T_t(\pi)$ and $T_t(\xi)$) converge weakly when t tends to ∞, then*

$$N(\xi) \leq_{st} N(\pi) \tag{6.1}$$

(resp.

$$T(\xi) \geq_{st} T(\pi)). \tag{6.2}$$

Observe that in steady state $T(\xi)$ and $T(\pi))$ are both equal to the arrival intensity owing to the ergodic theorem. However, the above result can be interpreted as stating that the stability region of the parallel system with ξ is larger than that associated with π.

Corollary 6.2 *If assumption A holds, and if the policies π and ξ defined in Proposition 4.3 are such that the random vectors $\vec{R}_n(\pi)$ and $\vec{R}_n(\xi)$ converge weakly when n tends to ∞, then*

$$R^i(\xi) \leq_{cx} R^i(\pi), \qquad i = 1, \cdots, |V|, \tag{6.3}$$

$$R(\xi) \leq_{cx} R(\pi). \tag{6.4}$$

Proof. For all convex function $f : I\!R \to I\!R$, $\sum_{n=1}^{N} f(R_n^i(\pi))$ is a symmetric convex function from $I\!R^N$ to $I\!R$. Proposition 4.1 together with Theorem 3.1 and the weak convergence assumptions yield

$$E[f(R^i(\xi))] = \lim_{N\to\infty} \frac{1}{N} \sum_{n=1}^{N} E[f(R_n^i(\xi))] \leq \lim_{N\to\infty} \frac{1}{N} \sum_{n=1}^{N} E[f(R_n^i(\pi))] = E[f(R^i(\pi))].$$

In a similar way, one can show

$$E[f(R(\xi))] \leq E[f(R(\pi))].$$

■

6.2 LOP in Steady State

The equivalence in distribution between the departure sequences of FCFS policies and those of LOP policies extends trivially from the transient regime to the stationary regime.

Corollary 6.3 *Assume that A holds and that policies θ and ξ defined in Proposition 5.4 are such that the random variables $N_t(\theta)$ and $N_t(\xi)$ (resp. $T_t(\theta)$ and $T_t(\xi)$) converge weakly when t tends to ∞, then*

$$N(\theta) =_{st} N(\xi) \tag{6.5}$$

(resp.

$$T(\theta) =_{st} T(\xi)). \tag{6.6}$$

6.3 SRF and LRF in Steady State

Now consider the stationary lag time of jobs. The Schur convex ordering result yields the following convex ordering result in steady state.

Corollary 6.4 *If assumption A holds, and if the policies σ, θ and λ defined in Proposition 5.5 are such that the random variables $L_n(\sigma)$, $L_n(\theta)$ and $L_n(\lambda)$ converge weakly when n tends to ∞, then*

$$\forall \theta \in \Theta,\ \exists \sigma \in \Sigma,\ \lambda \in \Lambda : \quad L(\sigma) \leq_{cx} L(\theta) \leq_{cx} L(\lambda). \tag{6.7}$$

Proof. The proof is similar to that of Corollary 6.2. ∎

When the information on the reference times is not used, the following extremal properties hold on the response times and lag times:

Corollary 6.5 *If assumption A holds, and if the policies ξ θ and υ defined in Proposition 5.6 are such that the random variables $R_n(\xi)$, $R_n(\theta)$ and $R_n(\upsilon)$ converge weakly when n tends to ∞. Then*

$$R(\xi) \leq_{cx} \quad R(\theta) \quad \leq_{cx} R(\upsilon). \tag{6.8}$$

Similarly, for all $\psi \in \Psi$, if the policies $\xi \in \Xi$ and $\upsilon \in \Upsilon$ are obtained from ψ by rearranging the service order so that FCFS and LCFS rules are satisfied, respectively, and if the random variables $R_n(\xi)$, $R_n(\psi)$ and $R_n(\upsilon)$ converge weakly when n tends to ∞. Then

$$L(\xi) \leq_{cx} \quad L(\psi) \quad \leq_{cx} L(\upsilon). \tag{6.9}$$

6.4 Rate of Successful Jobs

In term of the reference time, a job turns bad if its departure time is greater than its reference time, or equivalently, if the lag time is positive. A natural performance criterion in such systems is the rate of successful jobs: $P[L(\pi) \leq 0]$

Proposition 6.6 *If the relative reference times have a common concave distribution function, then under the assumptions of Corollary 6.5,*

$$P[L(\xi) \leq 0] \leq P[L(\psi) \leq 0] \leq P[L(\upsilon) \leq 0]. \tag{6.10}$$

Proof. Observe that for every policy ψ, $P[L(\psi) \leq 0] = P[R(\psi) \leq r]$ is a convex function of the stationary response time of job, $R(\psi)$, as the relative reference time r has a concave distribution. ∎

In words, LCFS (resp. FCFS) maximizes (resp. minimizes) the rate of successful jobs.

7 Preemptive Scheduling Policies

All the results obtained in the previous sections extend to the class of preemptive policies considered in this paper, i.e., an instance of a task can only be preempted by the instances of the same task. In place of Assumption A, we need the following one:

Assumption A': $\{s_{i,n}\}_{n=1}^{\infty}$, $1 \leq i \leq |V|$, *are mutually independent i.i.d. sequences of exponential RV's and are independent of* $\{u_n\}_{n=1}^{\infty}$, $\{r_n\}_{n=1}^{\infty}$.

Owing to the memoryless property of the exponential distribution, the proofs provided in this section still work if preemptive policies are considered. Therefore, all the results of the propositions and corollaries in this section hold when replacing assumption A and classes Π, Θ, Ξ, Υ, Σ, Λ, by assumption A' and classes Π', Θ', Ξ', Υ', Σ', Λ', respectively.

8 Randomly Structured Jobs

In the preceding sections, we assumed that the jobs have a fixed and identical task graph. We consider now the case where the task graphs have a common random structure, so that the jobs may have different task graphs. We will restricy ourself to the case where there is a finite set of possible graphs $\aleph = \{G(1) = (\mathcal{V}(1), \mathcal{E}(1)), \cdots, G(q) = \mathcal{V}(q), \mathcal{E}(q))\}$. Here, the graph G_n of the n-th job will be of type $G(i)$ with probability $p(i)$, where the $p(i)$'s sum up to one. The statistical assumptions will be the following;

Assumption B: *The sequence* $\{G_n\}_{n=1}^{\infty}$, *is i.i.d. and independent of* $\{s_{i,n}\}_{n=1}^{\infty}, \{u_n\}_{n=1}^{\infty}, \{r_n\}_{n=1}^{\infty}$.

As in the case of deterministic task graph, we assume that for all $i = 1, \cdots, q$, the mapping $\mathcal{M}(i) : \mathcal{V}(i) \rightarrow \{1, \cdots, K\}$ is fixed.

We are interested in determining whether the extremal properties established for FCFS, SRF, LRF, LOP, etc., policies still hold.

8.1 FCFS with Random Task Graphs

Generally speaking, the optimality properties of FCFS policies fail to hold when jobs have such a random structure. Consider the following example. Suppose there are two

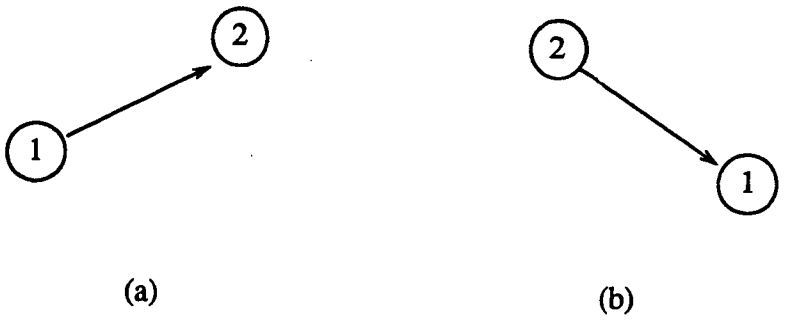


Figure 5: Counterexample of the optimality of FCFS policies with random task graphs. (a) The first job. (b) The second job.

machines. The first job arrives at time 0 and brings two tasks $T_{1,1}$ and $T_{2,1}$. The task graph is such that $T_{1,1} \rightarrow T_{2,1}$ (see Figure 5). The second one arrives at time, say 1, and brings two tasks $T_{1,2}$ and $T_{2,2}$. The task graph is such that $T_{2,2} \rightarrow T_{1,2}$ (see Figure 5). Suppose further that tasks $T_{1,1}, T_{1,2}$ are assigned to machine 1 and tasks $T_{2,1}, T_{2,2}$ are assigned to machine 2. In general, the FCFS policy, which requires $T_{2,1}$ be executed before $T_{2,2}$, is not optimal.

However the results obtained previously on FCFS policies extend to the class of modified FCFS policies, denoted by $\tilde{\Xi}$, where $\xi \in \tilde{\Xi}$ implies that

$$\forall m < n: \quad G_m = G_n \quad \Rightarrow \quad b_{i,m}(\xi) < b_{i,n}(\xi), \quad \forall i \in G_m.$$

The proof of this result is as in Section 4.

8.2 LOP with Random Task Graphs

Consider now the classes of LOP policies. In addition to the foregoing assumptions, we assume first that all the graphs $G(i)$, $i = 1, \cdots, q$, can be decomposed into a series of g subgraphs (such a series decomposition may not be minimal). Let G_n denote the sample graph of the n-th job. Denoting the series decomposition of graph G_n by $G_n^1, \cdots, G_n^g$, we also assume that

Assumption C: *The random graphs* $\{G_n^i,\ i = 1, g\}_{n=1}^{\infty}$ *are mutually independent, and for all* $1 \leq i \leq g$, *the sequence* $\{G_n^i\}_{n=1}^{\infty}$, *is i.i.d..*

It then follows that the corresponding subgraphs are interchangeable as needed in the

proofs of Section 5. Therefore, the Schur convex, convex symmetric and convex ordering properties concerning the class of LOP policies and its subclasses (e.g. SRF and LRF policies) hold for such random task graphs.

9 Applications to Specific Queueing Systems

It is worthwhile to observe that the results of the preceding sections do not provide sufficient information to identify the optimal policy. Indeed, simple combinatorial considerations show that classes of policies like FCFS, LCFS, SRF, LRF, etc., which contain the optimal policies, may contain a fairly large number of policies in addition to the optimal policy. However, in some specific cases, the optimal policy is uniquely determined. A particularly simple example is provided by the cases where two tasks of a same job are never assigned to the same machine. Then, the implementation of Local Order Preserving policies is much simpler, and each of the FCFS, LCFS, SRF and LRF classes of policies contain a single policy. The aim of the present section is to exemplify this particular situation.

9.1 $G/G/1$ Queues

The $G/G/1$ queue is a simple particular case of our system where there is only one machine and the job consists of a single task. It is clear that all policies are LOP: $\Pi = \Theta$, and that each of the FCFS, LCFS, SRF, and LRF classes of policies reduces to a single policy. The optimality results of section 6 on response times were already obtained in [5,8].

The results concerning lag times and the Schur convex ordering result of response times in transient regime, (see subsection 5.4 and subsection 5.5) are new, to the best of the authors knowledge.

9.2 Tandem Queueing Networks

Tandem queueing networks are another simple version of our general model. The task graph is a set of vertices in series. The number of machines is the same as the number of vertices in the task graph. Each task is allocated to a specific machine. In the minimal series representation, each subgraph is a single vertex. The virtual queues introduced in Section 5 for defining the series decomposition coincide with the tandem queueing network obtained by representing machines as servers and tasks as customers.

As in the $G/G/1$ queue, all policies are LOP: $\Pi = \Theta$, and the FCFS, LCFS, SRF, and LRF policies are uniquely defined. This gives rise to interesting applications in real time communications and to several new optimal schedulings results in the tandem queueing networks.

For instance, the results of the above section show that SRF (and LRF, respectively) at each queue minimizes (respectively, maximizes) any convex function of the stationary lag time, and that FIFO (and LIFO, respectively) at each queue minimizes (respectively, maximizes) any convex function of the stationary response time. For the lag times and the response times vectors in transient state, the Schur convex ordering results are also new.

9.3 Acyclic Fork/Join Queueing Networks

In [3], Acyclic Fork/Join queueing networks were defined and analyzed. This type of networks is also a special case of the general model presented in this paper. In such a network, there is one and only one task per job assigned to each machine.

Again, each of the classes of the policies described earlier defines a unique policy. The LOP policies require that the service orders at all the queues within each subgraph in the minimal series representation be the same.

One of the results that one can obtain for such a model is that FIFO at every queue maximizes stochastically the system throughput, and minimizes the convex functions of response time of each queue and of the system.

10 Conclusions

Our main results are summarized by the following relations. Assume A holds, and that the jobs have a deterministic task graph.

1. FCFS policies stochastically minimize job number and maximize throughput:

$$\forall \pi \in \Pi, \quad \exists \xi \in \Xi: \quad N_t(\xi) \leq_{st} N_t(\pi), \qquad \forall t > 0, \tag{10.1}$$

$$\forall \pi \in \Pi, \quad \exists \xi \in \Xi: \quad \mathcal{T}_t(\xi) \geq_{st} \mathcal{T}_t(\pi), \qquad \forall t > 0. \tag{10.2}$$

2. FCFS policies bound from below the vector of response times in the sense of Schur convex ordering:

$$\forall \pi \in \Pi,\ \exists \xi \in \Xi: \qquad \vec{R}(\xi) \leq_{scx} \vec{R}(\pi). \tag{10.3}$$

3. FCFS and LOP policies see identical (in law) departure sequences of jobs:

$$\forall \theta \in \Theta, \quad \exists \xi \in \Xi: \qquad N_t(\theta) =_{st} N_t(\xi), \qquad \forall t > 0, \tag{10.4}$$

$$\forall \theta \in \Theta, \quad \exists \xi \in \Xi: \qquad \mathcal{T}_t(\theta) =_{st} \mathcal{T}_t(\xi), \qquad \forall t > 0. \tag{10.5}$$

4. SRF and LRF policies bound from below and above, respectively, the vector of lag times, over the class of LOP policies, in the sense of Schur convex ordering:

$$\forall \theta \in \Theta,\ \exists \sigma \in \Sigma,\ \lambda \in \Lambda: \qquad \vec{L}(\sigma) \leq_{scx} \vec{L}(\theta) \leq_{scx} \vec{L}(\lambda). \tag{10.6}$$

5. FCFS and LCFS policies bound from below and above, respectively, the vector of response times, over the class of LOP policies, in the sense of Schur convex ordering:

$$\forall \theta \in \Theta,\ \exists \xi_i \in \Xi_i,\ \upsilon_i \in \Upsilon_i: \qquad \vec{R}(\xi_i) \leq_{scx} \vec{R}(\theta) \leq_{scx} \vec{R}(\upsilon_i). \tag{10.7}$$

6. The results concerned with transient state extend to the stationary state, provided it exists and it possesses certain ergodic properties. Namely,

$$\forall \pi \in \Pi, \quad \exists \xi \in \Xi: \qquad N(\xi) \leq_{st} N(\pi), \tag{10.8}$$

$$\forall \pi \in \Pi, \quad \exists \xi \in \Xi: \qquad \mathcal{T}(\xi) \geq_{st} \mathcal{T}(\pi), \tag{10.9}$$

$$\forall \theta \in \Theta, \quad \exists \xi \in \Xi: \qquad N(\theta) =_{st} N(\xi), \tag{10.10}$$

$$\forall \theta \in \Theta, \quad \exists \xi \in \Xi: \qquad \mathcal{T}(\theta) =_{st} \mathcal{T}(\xi) \tag{10.11}$$

The Schur convex ordering (which implies the symmetric convex ordering) on the transient random variables yields the convex ordering on the stationary random variables:

$$\forall \pi \in \Pi,\ \exists \xi \in \Xi: \qquad R^i(\xi) \leq_{cx} R^i(\pi), \quad i = 1, \cdots, |V|, \tag{10.12}$$

$$\forall \pi \in \Pi,\ \exists \xi \in \Xi: \qquad R(\xi) \leq_{cx} R(\pi), \tag{10.13}$$

$$\forall \theta \in \Theta,\ \exists \sigma \in \Sigma,\ \lambda \in \Lambda: \qquad L(\sigma) \leq_{cx} L(\theta) \leq_{cx} L(\lambda), \tag{10.14}$$

$$\forall \theta \in \Theta,\ \exists \xi \in \Xi,\ \upsilon \in \Upsilon: \qquad R(\xi) \leq_{cx} R(\theta) \leq_{cx} R(\upsilon). \tag{10.15}$$

7. If the task processing times have an exponential distribution or an Erlang distribution, or deterministic, then the above results hold for preemptive policies.

8. If the jobs have a common random structure, the optimality properties of the FCFS policies extend to the class of modified FCFS policies $\tilde{\Xi}$. Under the assumptions A and B, the extremal properties of SRF and LRF policies over LOP policies hold without any additional restriction.

In general, finding the optimal policy within this context is not a trivial task. Many additional combinatorial considerations have to be taken into account. However, if the processing system has a simple structure, e.g., acyclic fork join network or tandem network, the optimal policy can be fully determined.

Appendix

Algorithm for finding minimal series representation of an acyclic graph

$n = 1;$
$A^1 = B^1 = \{i \in V : p(i) = \emptyset\};$
$V^1 = B^1; W = V - B^1;$
while $W \neq \emptyset$ **do**
  if $\forall i \in B^n\ (s(i) = s(B^n))$
    then
      $G^n = (V^n, \{(i,j) : i,j \in V^n\});$
      $n = n + 1;$
      $A^n = B^n = V^n = s(B^{n-1});$
      $W = W - B^n;$
    else
      $w = \min_{v \in W} v; \quad W = W - \{w\};$
      $V^n = V^n + \{w\}; B^n = B^n + \{w\};$
      $B^n = B^n - p(w);$
  end if
end while
$G^n = (V^n, \{(i,j) : i,j \in V^n\})$

References

[1] F. Baccelli, Z. Liu, *On the Execution of Parallel Programs on Multiprocessor Architectures, a Queueing Theory Approach.* INRIA Report No. 833, April 88, also to appear in the J. ACM.

[2] F. Baccelli, Z. Liu, *Maximum Throughput of Parallel Executions.* Technical note CNET/PAA/ATR/SST/2239, August 1988, Issy, France, also submitted for publication to the IEEE Trans. on Computers.

[3] F. Baccelli, W. A. Massey, D. Towsley, *Acyclic Fork-Join Queueing Networks.* COINS Technical Report 87-40, March 1987, also to appear in the J. ACM.

[4] W. Clark, *The Gantt Chart*, (3rd Edition), Pitman and Sons, London, 1952.

[5] B. T. Doshi, E. H. Lipper, *Comparisons of Service Disciplines in a Queueing System with Delay Dependent Customer Behavior.* Applied Probability - Computer Science: The Interface, Vol. II, R. L. Disney, T.J. Ott, Eds., Cambridge, MA:Birkhauser, pp. 269-301, 1982.

[6] G. H. Hardy, J. E. Littlewood and G. Polya, *Some simple inequalities satisfied by convex functions.* Mess of Math. 58, 145, 152., 1929.

[7] G. H. Hardy, J. E. Littlewood and G. Polya, *Inequalities.* Cambridge University Press, 1934.

[8] J. G. Shantikumar, U. Sumita, *Convex Ordering of Sojourn Times in Single-Server Queues: Extremal Properties of FIFO and LIFO Service Disciplines.* J. Appl. Prob., Vol. 24, pp.737-748, 1987.

[9] D. Towsley, F. Baccelli, *Comparison of Service Disciplines in a Tandem Queueing Network with Delay Dependent Customer Behavior*, COINS Technical Report TR89-69, University of Massachusetts, July 1989, also submitted to Operations Research Letters.

[10] O. A. Vasicek, *An Inequality for the Variance of Waiting Time Under a General Queueing Discipline.* Operations Research, Vol. 25, pp. 879-884, 1977.

Imprimé en France
par
l'Institut National de Recherche en Informatique et en Automatique